SRM
Design Considerations

Arie Shoshani

Alex Sim

Junmin Gu

Andreas Mueller

Scientific Data Management Group

Computing Science Directorate

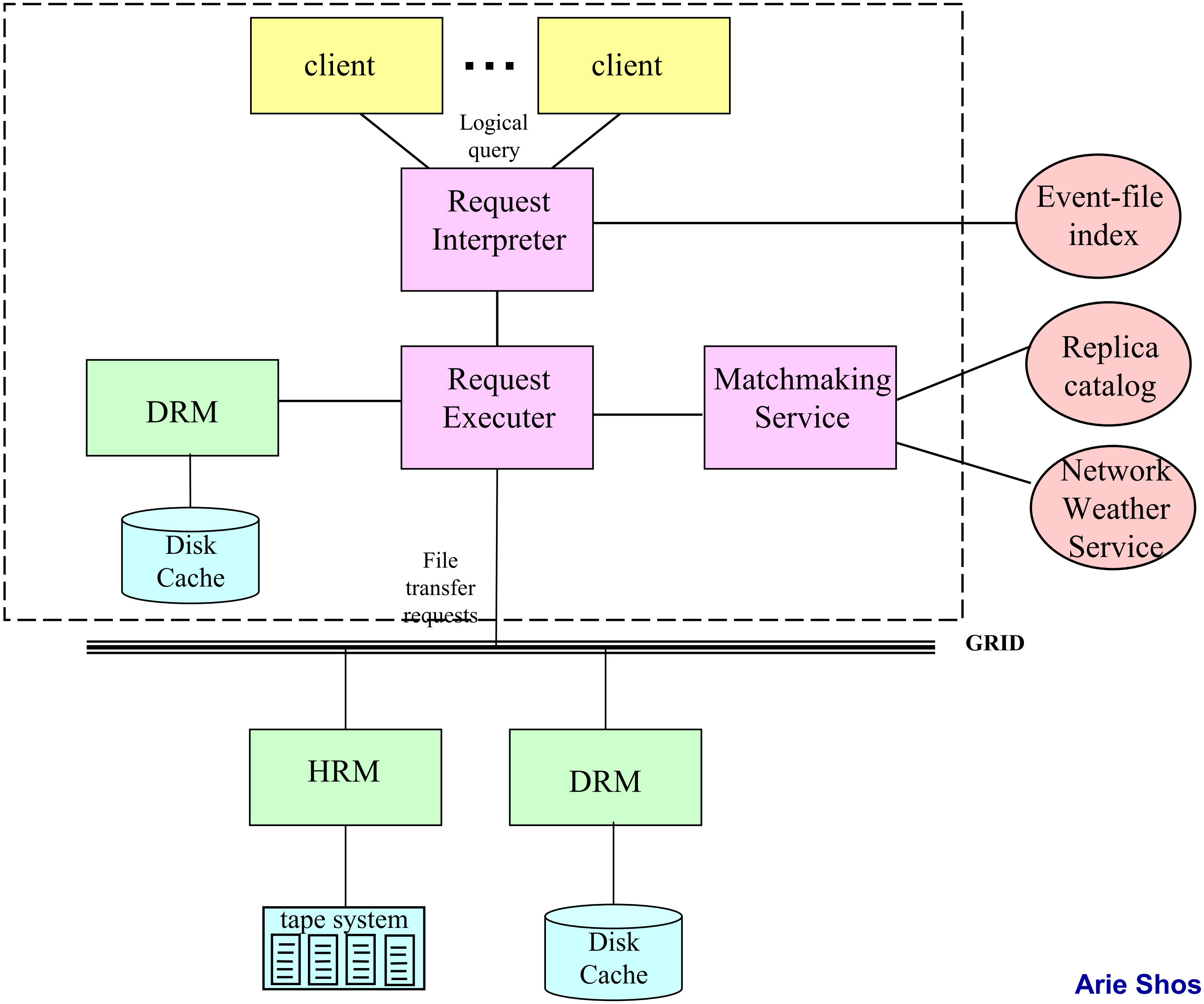
Lawrence Berkeley National Laboratory

June, 2001

Outline

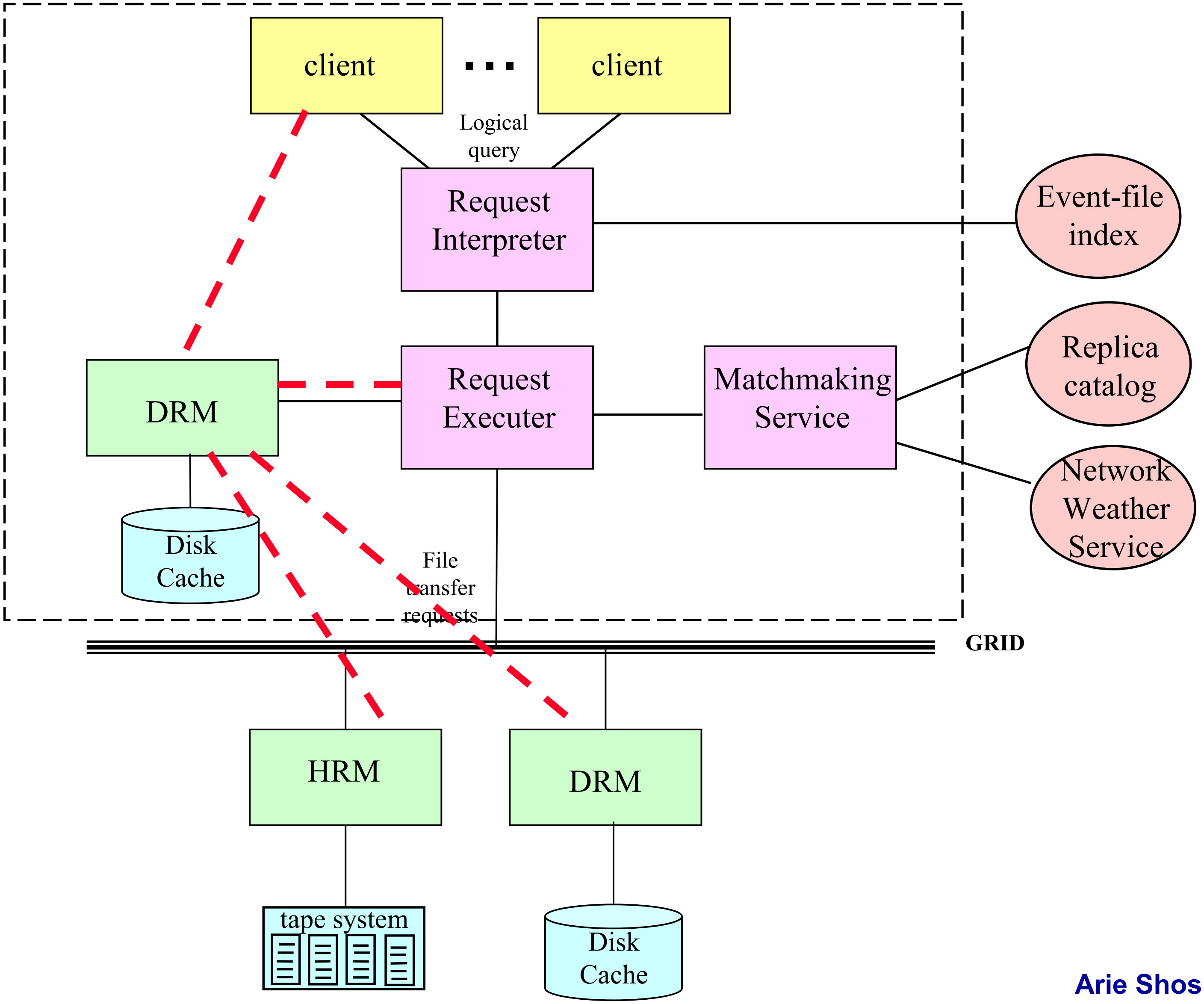
- Design Issues
  - Support clients directly? Or only through other agents (e.g. Request Manager)
  - Should SRMs get files if not in local cache?
  - Support push/pull?
  - Treat read/write separately?
  - Support a unified interface to DRM/HRM?
- Policies
  - Support for permanent / durable/ volatile files
  - Pinning level
  - User priorities
  - Run entire “job” (multiple file requests)
  - Notify RepCat of “volatile” staging? Support dynamic inquiry?
- Problems
  - How to control “overflow” writes?
  - How to make SRM robust (recover after crash)?
  - Pin-lock avoidance

Request Manager and SRMs



Arie Shoshani – June 2001

Communication with and between SRMs



Arie Shoshani – June 2001

Three scenarios that SRMs should be able to support

- A client communicates directly with DRM/HRM
  - No way to call client back
  - May ask for a local / remote file
- An agent calls DRM on behalf of a client
  - E.g. Request executer
  - It is possible to call agent back
  - May ask for local / remote file
- A DRM calls another DRM (or HRM)
  - As a result of a request for a remote file
  - To request a file

Should SRMs support clients directly?

- Yes, because:
  - Clients should be able to communicate directly to an SRM, not requiring special agents (such as ReqMgr)
  - e.g. running a simulation, writing to a DRM
  - e.g. running analysis, client knows files it wants
- Implications
  - Need to support “no_call_back” capability
  - i.e. support “status”
  - Client unreliable
    - Does not provide “release”
    - Does not provide “abort”
  - Therefore, need “time-out” mechanism support
  - Comment: “time-out” needed for all unreliable behavior, such a network crashes

Should SRMs get files if not in local cache?

- Yes, because:
  - Clients can communicate directly to an SRM
  - Does not require the architecture to have special agents (e.g. direct HRM-HRM replica support)
  - Allows DRM/HRMs to communicate directly with other DRM/HRM
- Implications
  - Provide logical_file_name + source URL for get/put
  - SRM returns local file URL
  - Support “call_back” and “status” (for simple clients)
- Benefit
  - Can design HRM as “DRM+TRM”

Should SRMs support push/pull?

- “Normal” behavior
  - — Get/pull, Put/push
- Problem
  - — unreliable behavior
    - Put/push gives “file size”, space allocated … writes more than “file size”
    - Get/pull is given “file size”, space allocated … gets more than “file size”
  - — How to detect?
    - Pull not a problem – can monitor transfer (policy: abort / get more space)
    - Push is a problem
- But, push is needed by clients “writes”
- Decision: support both “modes”
  - — Get/push useful for HRM = DRM+TRM

Treat read/write separately?

- Supprting "writes"
  - DRM: make space, perform pull/push
  - HRM: same as DRM + schedule put into tape
- Considerations
  - Separate queue for read and write
  - Separate space allocation for read and write
- Conclusion: no separation
  - No advantage to separate treatment
  - More complicated to implement
  - Priorities for write/read – a matter of policy

Support a unified interface to DRM/HRM?

- Yes, because:
  - Access to SRMs uniform
  - Simpler to implement
  - Staging performed “behind the scenes”
  - To the requester only the latency matters
    - HRMs can have a latency because of tape transfer and queues
    - DRMs can also have a latency - getting a file from another site (network transfer latency)
- Benefits
  - More uniform design
  - Clients communicate with DRMs and HRMs the same
  - DRMs + HRMs communicate uniformly
  - DRM can be used directly in HRM implementation

Interface Functionality

- Want to get a file
  - Request_to_get (push/pull)
  - Release
  - Abort
  - Status
  - Call_back (when file is available)
- Want to put a file
  - Request_to_put (push/pull)
  - Release
  - Abort
  - Status
  - Call_back_1 (when file is transferred to disk)
  - Call_back_2 (when file is transferred to tape – for HRM)